1 Quantum Shannon unitary decomposition

The algorithm for Quantum Shannon Decomposition (QSD) is a variation of the CSD algorithm presented in *cosine-sine unitary decomposition*. The change is made in the decomposition of the uniformly controlled operators L and R in the first identity of Figure 1 (explained in *cosine-sine unitary decomposition*). Being $U = U_0 \oplus U_1$ (as well as the matrices L and R of CSD), we can find the unitaries Q and W and the diagonal D that satisfy [1]

$$U = (I \otimes Q)(D \oplus D^\dagger)(I \otimes W). \tag{1}$$

From Equation (1), we have that $U_0 = QDW$ and $U_1 = QD^\dagger W$, which implies $U_0U_1^\dagger = QD^2Q^\dagger$. We then have that D and Q can be calculated through diagonalization. The unitary W can be found using the relationships $W = D^\dagger Q^\dagger U_0$ or $W = DQ^\dagger U_1$. Knowing that the diagonal D is composed of the eigenvalues of the diagonalization, the matrix $D \oplus D^\dagger$ can be represented by a uniformly controlled R_z rotation applied to the most significant qubit (Figure 2) [1].

Figure 1: Identities used for the cosine-sine decomposition (CSD) of quantum operators.

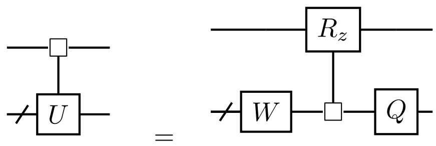

Figure 2: Identity used for quantum Shannon decomposition (QSD) of quantum operators.

Similarly to the CSD algorithm, the QSD applies the identities defined in Figure 1 and Figure 2 recursively. It starts by applying the first identity from Figure 1 to decompose the operator $U \in \mathbb{C}^{N\times N}$ into a central matrix D_{cs} and unitary matrices L_i and R_i. Next, the identity in Figure 2 is applied to the operators L and R, decomposing them into the diagonal $D \oplus D^\dagger$ and the $N/2 \times N/2$ dimensional unitaries W and Q (Figure 3). The procedure is restarted from the first identity, acting on the operators W and Q, and continues until these operators have a dimension of 2×2 representing one-qubit gates. The ZYZ decomposition used to represent one-qubit gates does not have CNOTs, thus only the uniformly controlled rotations contribute to the total number of CNOTs in the circuit $\frac{3}{4}4^n - \frac{3}{2}2^n$ [1].

If the recursion terminates when the operators Q and W act on two qubits, there will be 4^{n-2} such operators and $3\times 4^{n-2}$ fewer control-rotations by a qubit. Using the decomposition explained in *two-qubit unitary decomposition*, each two-qubit operator contributes with 3 CNOTs. Each control-rotation acting on two qubits contributes with 2 CNOTs [2]. Therefore, the CNOT count decreases by $3 \times 4^{n-2}$ and the total is reduced to $\frac{9}{16}4^n - \frac{3}{2}2^n$. This variant of the algorithm was called QSD ($l = 2$) by [1], where l indicates the number of qubits where the recursion terminates. The previous case, which terminates at one-qubit operators, is called QSD ($l = 1$).

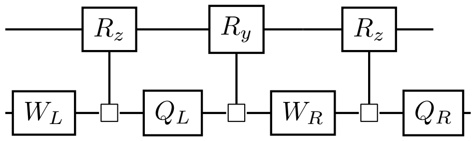

Figure 3: Circuito para a decomposição de Shannon (QSD) de um operador quântico de dois qubits.

The QSD ($l = 2$) can still be optimized in two ways [1]. The first, when the recursion ends with V and W operators acting on two qubits, applies the optimization indicated by Figure 4 and explained in *two-qubit unitary decomposition*, reducing $4^{n-2} - 1$ CNOTs (one CNOT per two-qubit operator, except for the last one). The second optimization uses CZ (Controlled-Z) gates instead of CNOTs to decompose the central matrices, as the last CZ can be absorbed by the neighboring multiplexer, saving $(4^{n-2} - 1)/3$ CNOTs. With both optimizations, the total number of CNOTs is reduced to $\frac{23}{48}4^n - \frac{3}{2}2^n + \frac{4}{3}$. So far, this is the most efficient decomposition for general unitaries. [1] referred to this optimized decomposition as QSD ($l = 2$, optimized).

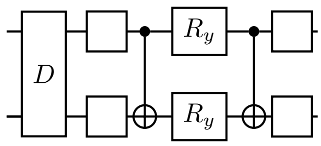

Figure 4: Diagonalization of arbitrary two-qubit unitaries.

Listing 1: Quantum Shannon decomposition ($l = 2$, optimized).

```
from qiskit import transpile
from scipy.stats import unitary_group
from qclib.unitary import unitary as decompose
table = '|␣n␣|␣cnots␣␣|␣depth␣␣|\n'
table += '|:-:|:------:|:------:|\n'
for n_qubits in range(3, 9):
    U = unitary_group.rvs(2 ** n_qubits)
    circuit = decompose(U, decomposition='qsd')
    t_circuit = transpile(circuit, basis_gates=['u', 'cx'],
                                optimization_level=0)
    n_depth = t_circuit.depth()
    n_cx = t_circuit.count_ops().get('cx', 0)
    table += f'|␣{n_qubits}␣|␣{n_cx:^6d}␣|␣{n_depth:^6d}␣|\n'
print(table)
# | n | cnots  | depth  |
# |:-:|:------:|:------:|
# | 3 |   20   |   45   |
# | 4 |  100   |  225   |
# | 5 |  444   |  1009  |
# | 6 |  1868  |  4273  |
# | 7 |  7660  | 17585  |
# | 8 | 31020  | 71345  |
```

References

[1] V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6):1000–1010, 2006.

[2] Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. Quantum circuits for isometries. Physical Review A, 93(3):032318, 2016.